Dynamic Filtering of Useless Data in an Adaptive Multi-Agent System: Evaluation in the Ambient Domain

Valérian Guivarch, Valérie Camps, André Péninou, and Simon Stuker

Institut de Recherche en Informatique de Toulouse
{Valerian.Guivarch,Valerie.Camps,Andre.Peninou,Simon.Stuker}@irit.fr
http://www.irit.fr

Abstract. *Amadeus* is an Adaptive Multi-Agent System whose goal is to observe and to learn users' behaviour in an ambient system in order to perform their recurrent actions on their behalf. Considering the large number of devices (data sources) that generally compose ambient systems, performing an efficient learning in such a domain requires filtering useless data. This paper focuses on an extended version of *Amadeus* taking account this requirement and proposes a solution based on cooperative interactions between the different agents composing *Amadeus*. An evaluation of the performances of our system as well as the encouraging obtained results are then shown.

Keywords: Adaptation, Learning, Distributed-problem solving, Data filtering, Pervasive agents, Ambient intelligence.

1 Introduction

The performances of learning algorithms is generally degraded by the presence of useless data among the ones used to learn, a piece of data being considered to be useless if there is no link between its value and the objective to learn. One way to improve this fact is to select useful data. An ambient system is composed of many heterogeneous devices, often mobile, physically distributed and interacting in a dynamic way. So, it is a typical example where applying a learning is a very complex task, because it consists of a great number of devices that are able to generate data. Furthermore, devices may appear and disappear dynamically. Thus, in this case, the filtering of data coming from these devices cannot be defined *a priori*, that is before the learning process. Learning has to be done at runtime, without restarting from scratch when new data appear. The filtering of useless data has also to be done at runtime.

This is particularly the case for the multi-agent system *Amadeus* [4] whose goal is, in an ambient system, to observe the users' actions in order to learn those that are recurrent and to learn then how to perform them on behalf of the users. This learning is performed in a decentralized way, an instance of *Amadeus* being responsible of each device of the ambient system. However, the large number of

devices requires the filtering of useless data at runtime for the users' behaviour learning.

In this paper, we present an extended version of *Amadeus*. Section 2 briefly presents the general functioning of *Amadeus*. Our contribution enabling the "on-line" filtering of useless data is presented in section 3 and evaluated in section 4. Section 5 is devoted to related works. Section 6 concludes this paper and explains the on-going work.

2 Our MAS Proposition for Ambient System: *Amadeus*

Our contribution aims at proposing a solution to tackle the problem of adaptation in ambient systems. We propose to make an ambient system able to provide a relevant behaviour, based on the perceived user's actions, in order to assist him by realizing his actions on his behalf. We have already proposed *Amadeus* [4], an Adaptive Multi-Agent System that is able to learn the user's contexts while this user is performing recurrent actions in order to act on his behalf in similar situations.

An instance of the *Amadeus* multi-agent system is associated to each device of an ambient system. Figure 1 is a representation of an instance of *Amadeus*. We can observe the four types of agents existing in the system: *Data* agents, *User* agents, *Context* agents and *Controller* agents.

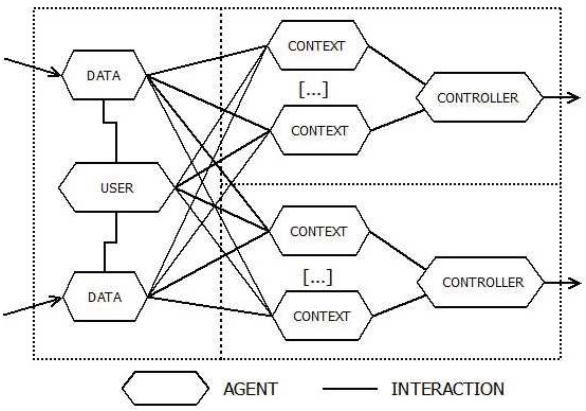

Fig. 1. Representation of an instance of *Amadeus*, one instance is associated to each device of an ambient system

A *Controller* agent is associated to each effector of a device. Its goal is to decide at anytime what is the best action to perform on the effector on behalf of the user. This decision is made thanks to a set of *Context* agents.

A *Context* agent is created each time a user performs an action in his environment (for example to turn on the light). This agent associates this action with a description of the situation in which the user performs this action. This situation is composed of the set of the perceived data values when the action is performed (example: Light = 0 ; PresenceSensor= 1 ; LuminositySensor = 22).

The action is represented by the value given to the effector (for example, 1 to turn on and 0 to turn off the light).

The *Context* agent adds to this description a forecast on the effect of this action on the user's satisfaction level (does the completion of this action for this situation lead to increase, to maintain or to decrease the user's level satisfaction?). This forecast is obtained by comparing the user's satisfaction level before and after the achievement of the action. This satisfaction is represented by a value between 0 (the user is not satisfied at all) and 1 (he is very satisfied), and is estimated by the *User* agent. The *User* agent is responsible for the user's preferences. Thanks to a representation of the user's preferences (currently with a XML file) the *User* agent can evaluate, for any situation, if the user is satisfied with regard to the state of the device effector. One of our work perspectives is to make the *User* agent able to dynamically and autonomously learn these preferences, but this is out of the scope of this paper.

Every *Context* agent perceives a set of data coming from local sensors situated on either the same device or on distant sensors situated on other instances of *Amadeus* (namely on devices). The *Data* agents represent these data. In the *Context* agent, each of these data possesses a validity status that depends on its current value with regard to the situation described (by the *Context* agent). A data is considered as valid if it is included in a values range. This range represents the values interval that a piece of data may have in order to describe a situation. Thus, the *Context* agent tries to establish the borders of valuable ranges for every perceived data that enable it to describe the correct situation for which its action proposition is appropriate (namely it will have the expected effects). To do this, the *Context* agent possesses, for each perceived data, an Adaptive Range Tracker (ART) that is a data structure enabling to describe a valuable interval (called "interval of validity") where min and max borders can evolve. The value of each border is estimated with an Adaptive Value Tracker (AVT) [11], which is a tool devoted to the tuning of evolving parameters. A *Context* agent considers a data as valid if its current value belongs to its associated ART. A *Context* agent has also its own validity status. Its status is valid if all perceived data are valid (it is invalid otherwise). In this case, a *Context* agent sends its action proposition and its forecast to the *Controller* agent. The *Controller* agent can then decide which action, among those proposed by all valid *Context* agents, is the most appropriate to satisfy the user.

A first evaluation of this *Amadeus* version applied to a simple example gave us encouraging results [4] [5]. Nevertheless, we observed a strong weakness regarding the required learning time when the number of perceived data increases. In particular, the addition of "useless" data that change independently of the user's actions on an effector, implies a strong slowing down of the learning time. Such data are perceived by *Context* agents, and so are included in the situation description, but they do not affect the user' behaviour. For example, *Context* agents can perceive humidity level of the rooms but it is not necessary to consider this fact to decide to turn on or to turn off the lights. Indeed, if a situation having previously led to the realization of an action by the user appears again,

with useless piece of data having a different status, the system considers wrongly that the situation is different. For example, if the system has learnt to switch on the light when the user goes into the room but with a specific level of humidity, when the user enters again into the room, a change of humidity level perceived by a humidity sensor leads the system to consider itself in a new situation, and thus to not act.

To overcome this problem, it is necessary to determine, for each effector, which data are useless for learning the behaviour to give to this effector, in order to only consider the useful data. Describing explicitly what are the useful and the useless data for each effector seems to be a limited solution, because of the strong dynamic of an ambient system. This is why we propose to make each instance of *Amadeus* able to autonomously determine, without any *a priori* information and at runtime, what are the data useless to learn the user's behaviour.

3 Data Filtering

The objective of our study is to locally detect what are the useless data for a device. A piece of data is considered as useless if its value is independent of the user's decision to act on this device. This detection is based on a learning performed at runtime, without any *a priori* information.

3.1 Proposed Approach

Our proposition to detect useless data is based on the cooperation between agents. On the one hand, the *Context* and *Controller* agents are responsible for the user's behaviour learning towards the state of an effector. On the other hand, the *Data* agents are responsible for providing useful data to *Context* and *Controller* agents so that they make their learning. Thus, *Data* agents and *Context* agents have to interact for determining which data are useful to characterise the situation in which the user acts on the device, and which data are useless.

A *Context* agent is created each time the user performs an action on an effector. This agent associates the user's action with a description of the situation in which the user had performed it (see section 2). This situation is composed of the set of the perceived data values when the action was performed. The assumption is that all perceived data contribute to characterize the situation. Thus, later, when the *Context* agent becomes valid, it can be sure of its decision. But what about when it is not valid? Is it invalid because all data contributes really to describe the situation (it is right to be invalid) or because one useless data possesses a current value making the agent invalid (it is wrong to be invalid)? The *Context* agent cannot solve this ambiguity by itself. Nevertheless, a more interesting point of view is to think about useful and useless data when at once i) the *Context* agent is invalid and ii) another one is valid. In such cases, the *Context* agent can question itself about the necessity to be invalid in the current situation. We use these cases in our approach described hereafter.

When a *Context* agent is selected and its action is made by the *Controller* agent (or when the user makes an action himself), every invalid *Context* agent

observes its internal state, and evaluates if it was right to do not send its action proposition. More details on this evaluation are given in section 3.2. When a *Context* agent establishes it was right to be invalid at a given time, it was thanks to the invalid perceived data. So, among these currently invalid data, it knows that there is at least one useful piece of data. However, without more information, the *Context* agent cannot determine if a data is invalid because it is really useful, or if it is just a coincidence that this data is invalid in the current situation. So, it sends a message called a "*usefulness signal*" to all the data that are invalid in order to inform them that they are potentially useful.

In order to evaluate the usefulness of its data, each *Data* agent perceives the *usefulness signals* sent by all the *Context* agents. The goal of a *Data* agent is to process these signals in order to establish if the reception of these usefulness signals is just the result of coincidences (the associated data was invalid at the good moment, but even if it had not been, a data really useful would have been invalid) or if it was invalid at the good moment because it is a useful data. This process is described in section 3.3.

3.2 Usefulness Signal Generation

Let us consider a *Context* agent C. Another *Context* agent S has just been selected, whereas C was invalid, because among the set of perceived data D, there is a subset of invalid data D_I for C. So, agent C observes the selected agent S to know if it proposed the same action or not, and if this proposition was or not associated with the same forecast on the effect of this action on the user's satisfaction level. Two particular cases can be highlighted.

The first one occurs when C and S propose the same action (for example, turn on the light) but with different forecasts. It is obvious that the same action cannot have two different effects in the same situation. Then, the *Context* agent C was right to be invalid when S was valid. So, at least one of the invalid data of C is useful. Let us give an example for this case. S proposes to turn on the light when someone is in the room with a low luminosity level and a high humidity level. On the contrary, C proposes to turn on the light when someone is in the room with a high luminosity level and a low humidity level. S forecasts an increase of the user's satisfaction if it is selected, whereas C forecasts a decrease of the user's satisfaction if it is selected (to turn on the light if the luminosity is high is not satisfying for the user). When S is selected, the situation is: someone is in the room, low luminosity level and high humidity level. C observes that it proposes the same action as S, but with a different forecast. So it can consider that in the set of invalid data composed by the luminosity and the humidity levels, there is at least one useful data for it (in this case, the luminosity data).

The second case occurs when C and S propose two different actions with the same forecast. So, if S proposes an action that increases the user's satisfaction level, C cannot be valid in the same situation if it proposes a different action that also increases the user's satisfaction level. This is why C can consider that, in the set of its invalid data, at least one is useful. Let us give an example for this case. S proposes to increase the user's satisfaction level by turning on the light

when the user is in the room with a low luminosity level and a high humidity level. On the contrary, C proposes to increase the user's satisfaction level by turning off the light when the user is in the room with a high luminosity level and a low humidity level. When S is selected, the situation is: the user is in the room, low luminosity level and high humidity level. C evaluates that it cannot be valid at the same time, so it considers that, in the set of its invalid data composed by the luminosity level and the humidity level, there is at least one useful data.

For each of these situations, C sends a *usefulness signal* to each of its invalid data in order to warn them that they are "maybe" useful.

To conclude, *Context* agents are able to detect and gather situations or information about the usefulness of perceived data. However, these information concern the usefulness of data, whereas we are interested in their uselessness. So, in the next section, we describe how the *Data* agents process the *usefulness signal* to detect if they are useful data or not.

3.3 Usefulness Signal Processing

First of all, let us underline that *Context* agents and *Controller* agents are always bound to a single effector of a device (a complex device may have different effectors). *Data* agents receive *usefulness signals* that are implicitly bounded to an effector. Thus, *Data* agent processing of these signals must be performed separately for each effector.

We consider two data F and L, where F is useful and L is useless with respect to an effector E. Also, we consider two sets of *Context* agents $SC1$ and $SC2$, where the agents of $SC1$ propose to switch the effector to some state e_1, and the agents of $SC2$ propose to switch the effector to some other state e_2. Each time a *Data* agents (F or L) receives a *usefulness signal*, it observes its current value and the state of the effector E. With these values, it computes a density function of the values taken at the reception of *usefulness signals* regarding the effector state proposed by the *Context* agent that sent the signal. Let S_E denotes the set of possible states of the effector E. So, for each state e in S_E, and each *Data* agent D, $d_D(e)$ denotes the density function of agent D with respect to effector's state e when *usefulness signals* are received.

The distinction between the useful data F and the useless data L can be observed through the density functions $d_F(e)$ and $d_L(e)$. Because *Data* agent F is useful, it is correlated with the actions applied on the effector, hence $d_F(e_1) \neq d_F(e_2)$. Conversely, *Data* agent L is useless and has no influence on the effector's actions. We can observe this fact through the similarity between $d_L(e_1)$ and $d_L(e_2)$.

On the basis of these remarks, we can evaluate the usefulness of a *Data* agent by comparing the density functions corresponding to the different states of the effector. More precisely, the distance between two density functions $d_D(e_1)$ and $d_D(e_2)$ is measured through the *Chi-square distance* [3] relative to the general data frequency:

$$\delta\left(d_D(e_1), d_D(e_2)\right) = \sum_{Value\ i} \frac{(d_i - d'_i)^2}{d'_i}$$

where d_i (respectively d'_i) denotes the frequency of value i for the *Data* agent D (whenever D is considered useful by *Context* agents) with respect to the effectors state e_1 (respectively e_2). The usefulness of a *Data* agent D, $U_D(E)$, can then be expressed as the maximum distance between any pair of its density functions:

$$U_D(E) = \max_{x,y \in S_E} \left(\delta\left(d_D(x), d_D(y)\right) \right)$$

The use of the *chi-square distance* in order to compare two density functions allows obtaining a value that has a statistical significance. As a matter of fact, under the assumption that the data is useful, $\delta\left(d_D(e_1), d_D(e_2)\right)$ follows some *chi-square* law [1]. So, we can grant a statistical credibility to the evaluation of the usefulness data.

When a *Data* agent receives a *usefulness signal*, it computes its density functions with respect to the effector states. So, the evaluation of its usefulness based on these density functions becomes more and more precise. Finally, when the usefulness level of a *Data* agent gets below a certain fixed threshold empirically calculated, the *Data* agent considers that it is useless for the effector.

In this case, the *Data* agent informs the set of *Context* agents associated to this effector of its uselessness, and then its stops to send them update about its value. Then the *Context* agents delete the ART associated with this data, and forget this data to estimate their validity state.

Our filtering is based on the signal sent by the *Context* agents, each signal concerning a set of *Data* agents. So, contrary to classical methods that filter each data independentely, this method takes care about dependencies between data.

4 Experimentations

The proposed solution was implemented using Speadl/MAY [12], which is a tool allowing to assemble reusable components in order to build architectures supporting the development and execution of multi-agent systems. Our solution was evaluated through a simulator using users' preferences (given in a XML file) to generate users' behaviour in a virtual ambient system.

Our experimentation takes place in a case study composed of an apartment with five rooms (one living room, one bathroom, one kitchen and two bedrooms). Each room possesses a light, a luminosity sensor and a presence sensor. Each *Amadeus* instance associated to a light effector has to learn a good behaviour based on the user's actions on this effector. Each instance perceives not only the data from its room (the luminosity sensor, the presence sensor, and the state of the light), but also the data of the other rooms. Among the fifteen perceived

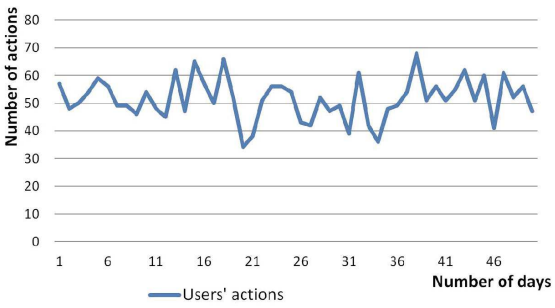

Fig. 2. Number of users' actions per day without *Amadeus*

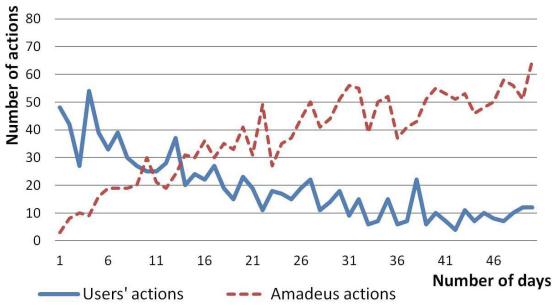

Fig. 3. Number of users' actions and *Amadeus* actions per day without filtering data

data (three by room and five rooms), only three data are useful for each instance, and twelve are useless.

We added three users to this simulation. These users can move between the different rooms; they can also leave (and come back to) the apartment (we consider a sixth room called "outside" without any sensor and effector). The users' behaviour is based on simple rules: when a user enters in a room with a switched off light and an insufficient luminosity, he turns on the light, whereas if the luminosity is very strong with a switched on light, the user turns off the light. When he leaves a room, if he was the last in this room and the light was switched on, he turns off the light. Figure 3 represents the number of users' actions per day during a simulation of fifty days, without the *Amadeus* use. We can observe an average number of 50 actions per day made by the different users.

The first experiment consists in adding an instance of *Amadeus* (first version without useless data filtering) to each device of our case study. Every instance associated with the light device is in charge to learn the correct behaviour to give to this light device according to the users' actions. Figure 3 represents the number of actions respectively made by the users and *Amadeus* on the different devices during a simulation of fifty days. We can observe that, even if *Amadeus* makes many actions on behalf of the users, its performances are very limited and the users have to make a lot of actions even after 50 learning days. This can be explained by the fact that each *Amadeus* instance responsible for each light device has difficulties to learn in which situation every action is realized, because of the too numerous useless perceived data.

The second experiment consists in carrying out the same experiment with *Amadeus* instances able to filter useless data. So, based on the process explained in section 3, the *Data* agents are able to locally and autonomously detect what are the useless perceived data for each device. For each of the five effectors, 12 useless data have to be filtered (hence a total number of 60 useless data for the entire system). After fifty days of simulation, the system has filtered 48 useless data (11 for the first light, 9 for the second, 8 for he third, 9 for the fourth and 11 for the fifth). Figure 4 shows the *Amadeus* capabilities to make actions on behalf of the users with the use of filtering useless data, only 15 days being necessary to decrease users' actions to less than 10 per day.

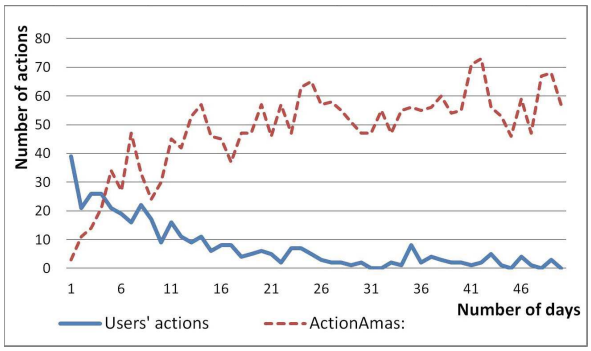

Fig. 4. Number of users' actions and *Amadeus* actions per day with filtering data

This simulation has been made twenty times, in order to evaluate the quality of the filtering of useless data on several simulations. Each simulation gives a quite random behavior to each user for moving in the apartment. Table 1 displays obtained results. The first line shows the total number of filtered data whereas the second line shows the number of useful data wrongly filtered. The third and fourth lines respectively show the percentage of useless data filtered, and the percentage of useful data wrongly filtered. We obtained a final average percentage of useless data filtered equal to 79.7, and an average percentage of useful data wrongly filtered equal to 1.

Table 1. Results of the evaluation of *Amadeus* for twenty simulations

Total number of filtered data	55	42	51	41	48	52	46	56	50	49
Number of useful data wrongly filtered	0	0	0	0	0	0	0	0	0	1
Percentage of useless data filtered	91,7	70	85	68,3	80	86,7	76,7	93,3	83,3	81,7
Percentage of useful data wrongly filtered	0	0	0	0	0	0	0	0	0	6,7

Total number of filtered data	48	38	42	45	49	49	50	49	47	49
Number of useful data wrongly filtered	0	0	0	0	0	1	0	0	1	0
Percentage of useless data filtered	80	63,3	70	75	81,7	81,7	83,3	81,7	78,3	81,7
Percentage of useful data wrongly filtered	0	0	0	0	0	6,7	0	0	6,7	0

The data selection performed by the *Data* agents is not perfect because about 20% of useless data remain unfiltered. However, it is possible to decrease the chosen threshold to decide data uselessness in order to have better results regarding the data filtering, but it would imply an increase of the rate of wrong filtering. Nevertheless, our objective is not necessary to filter all the useless data, but to filter a sufficient number of useless data in order to make *Amadeus* able to learn the users' behaviour. Hence, we consider that it is better to have unfiltered useless data than wrongly filtered useful data.

5 Related Work

A very explicit illustration of the useless data effect on learning algorithms is given by [2]. Figure 5 shows an example where a learning algorithm tries to classify information into two classes. If the learning algorithm observes the A and B points using only the data x_1 (represented by their projection A' and B' on the x_1 axis), it will find correctly the two classes (dotted vertical line). However, if it considers the useless data x_2, it will fail to separate the two classes (diagonally continuous line). So, considering useless data in the learning process makes it necessary to increase the number of examples that must be provided to the learning algorithm in order to overcome this problem.

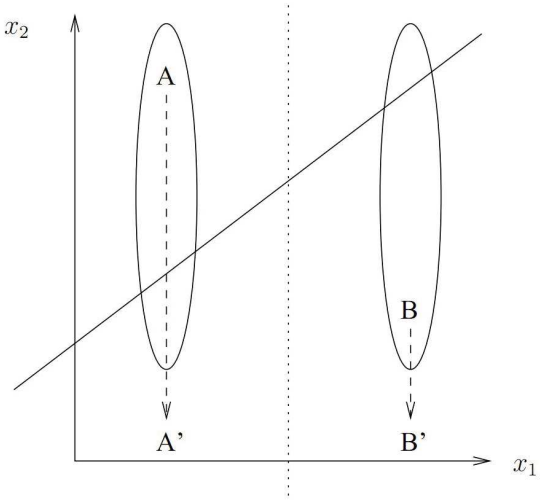

Fig. 5. Illustration of the effect of useless data on a learning algorithm (from [2])

In the literature, many solutions were proposed to solve this very recurrent problem in learning methods [7]. A first solution concerns variable ranking methods [6]. Such methods try to compute a score for each data that represents the usefulness of this data with respect to the target of the learning. This evaluation is performed independently of each data, and data with a low score are considered as useless. However, in complex systems such as ambient systems, the large number of interactions between data makes such methods inappropriate. As a matter of fact, the effect of a piece of data on the target of the learning can be strongly dependant on other data, so an evaluation of the usefulness of this piece of data independently of other data is not appropriate [14]. For example, in the study of the section 4, the usefulness of luminosity sensor is strongly depending of the value of presence sensor, so it is necessary to evaluate the usefulness to this data regarding the values of other data.

Other methods allow the selection of useful data by considering subsets of data rather than independent pieces of data. These methods can be divided into three categories [13]:

1. *Filter* techniques that carry out the selection of useful data independently of the learning algorithm itself; they are unable to interact with this one during learning process [9].
2. *Wrapper* techniques that use the learning algorithm itself in order to evaluate the usefulness of subsets of data. For this, the learning algorithm is applied to different subsets of data, and the performances of the learning are evaluated regarding the data subset to which it is applied. They generally get better results than *Filter* techniques, but they require much more computation time and they present a strong risk of overfitting [8].
3. *Embedded* techniques that are directly integrated in the learning algorithm itself (the variable selection being strongly correlated with the process of the learning algorithm). For example, some algorithms observe the impact of the addition or the removal of data in their learning process in order to evaluate the usefulness of these data [10].

Because of the dynamic and distributed properties of the learning algorithm implemented by *Amadeus*, we considered, as we implemented it, that an embedded technique of data selection was the most appropriate solution.

6 Conclusion

In this paper, we have presented an extended version of the multi-agent system *Amadeus*. This system is devoted to the learning of users' behaviour in ambient systems. *Amadeus* can learn situations and actions performed by the user on effectors of devices in order to perform later these actions on behalf of the user. However, the large number of useless data, because of the high number of devices in ambient systems, makes necessary to improve *Amadeus* learning by adding filtering data capabilities. We have introduced a new ability to the *Data* agents, which purpose is to detect if a piece of data is useless for every effector of a device in the ambient system or not. We also described the data filtering process performed by the *Data* agents, and presented first results about the *Data* agents' filtering performances.

The choice to define a threshold to decide of the uselessness of each data gives encouraging results. For example, *Amadeus* is able to filter a large part of useless data, and it wrongly filters a very low level of useful data. However, in an adaptive multi-agent system, having a static parameter is a weakness in term of real adaptation. That is why we are currently investigating a solution to make the *Data* agents able to filter useless data and to autonomously and dynamically define the proper threshold to use.

References

1. Baillargeon, G.: Introduction à l'inférence statistique: méthodes d'échantillonnage, estimation, tests d'hypothèses, corrélation linéaire, droite de régression et test du khi-deux avec applications diverses. Techniques statistiques. Les Editions SMG (1982)

2. Delalleau, O.: Extraction hiérarchique de caractéristiques pour l'apprentissage à partir de données complexes en haute dimension, Pre-doctoral report, University of Montreal (2008)
3. Greenwood, P.E., Nikulin, M.S.: A Guide to Chi-Squared Testing. Wiley, New York (1996)
4. Guivarch, V., Camps, V., Péninou, A.: Context awareness in ambient systems by an adaptive multi-agent approach. In: Paternò, F., de Ruyter, B., Markopoulos, P., Santoro, C., van Loenen, E., Luyten, K. (eds.) AmI 2012. LNCS, vol. 7683, pp. 129–144. Springer, Heidelberg (2012)
5. Guivarch, V., Francisco De Paz Santana, J., Bajo Pérez, J., Péninou, A., Camps, V.: Learning user's behaviour with an Adaptive Multi-Agent System approach (regular paper). In: Intelligent Systems for Context-based Information Fusion, Cartagena de Indias - Colombia, Springer (2012)
6. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research 3, 1157–1182 (2003)
7. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A.: Feature extraction: foundations and applications, vol. 207. Springer (2006)
8. John, G.H., Kohavi, R., Pfleger, K., et al.: Irrelevant features and the subset selection problem. In: Proceedings of the Eleventh International Conference on Machine Learning, San Francisco, vol. 129, pp. 121–129 (1994)
9. Kira, K., Rendell, L.A.: The feature selection problem: Traditional methods and a new algorithm. In: Proceedings of the National Conference on Artificial Intelligence, p. 129. John Wiley & Sons Ltd. (1992)
10. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
11. Lemouzy, S., Camps, V., Glize, P.: Principles and Properties of a MAS Learning Algorithm: a Comparison with Standard Learning Algorithms Applied to Implicit Feedback Assessment (regular paper). In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT), Lyon, pp. 228–235. CPS (Conference Publishing Services) (août 2011)
12. Noel, V.: Component-based Software Architectures and Multi-Agent Systems: Mutual and Complementary Contributions for Supporting Software Development. Thése de doctorat, Université de Toulouse, Toulouse, France (juillet 2012)
13. Saeys, Y., Inza, I., Larrañaga, P.: A review of feature selection techniques in bioinformatics. Bioinformatics 23(19), 2507–2517 (2007)
14. Shafti, L.S., Haya, P.A., García-Herranz, M., Pérez, E.: Evolutionary feature extraction to infer behavioral patterns in ambient intelligence. In: Paternò, F., de Ruyter, B., Markopoulos, P., Santoro, C., van Loenen, E., Luyten, K. (eds.) AmI 2012. LNCS, vol. 7683, pp. 256–271. Springer, Heidelberg (2012)